BACHELOR OF

Engineering & Technology



SCHOOL OF COMPUTING SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA

UTTAR PRADESH

**SUBMITTED TO :–**

**SUBMITTED BY:-**

**SECTION -**

***BRESENHAM’S ALGORITHM FOR LINE DRAWING.***

1. Start.
2. Declare variables x,y,x1,y1,x2,y2,p,dx,dy and also declare gdriver=DETECT,gmode.
3. Initialize the graphic mode with the path location in TC folder.
4. Input the two line end-points and store the left end-points in (x1,y1).
5. Load (x1,y1) into the frame buffer; that is, plot the first point put x=x1,y=y1.
6. Calculate dx=x2-x1 and dy=y2-y1,and obtain the initial value of decision parameter p as:
   1. p=(2dy-dx).
7. Starting from first point (x,y) perform the following test:
8. Repeat step 9 while(x<=x2).
9. If p<0,next point is (x+1,y) and p=(p+2dy).
10. Otherwise, the next point to plot is (x+1,y+1) and p=(p+2dy-2dx).
11. Place pixels using putpixel at points (x,y) in specified colour.
12. Close Graph.
13. Stop.

**WAP TO DRAW A LINE USING MID POINT ALGORITHM OR BRESENHAM’S ALGORITHM.**

#include<stdio.h> #include<conio.h> #include<graphics.h> void main()

{

int x,y,x1,y1,x2,y2,p,dx,dy; int gdriver=DETECT,gmode;

initgraph(&gdriver,&gmode,"C:\\tc\\BGI:"); printf("\nEnter the x-coordinate of the first point ::"); scanf("%d",&x1);

printf("\nEnter the y-coordinate of the first point ::"); scanf("%d",&y1);

printf("\nEnter the x-coordinate of the second point ::"); scanf("%d",&x2);

printf("\nEnter the y-coordinate of the second point ::"); scanf("%d",&y2);

x=x1; y=y1; dx=x2-x1; dy=y2-y1;

putpixel(x,y,2);

p=(2dy-dx); while(x<=x2)

{

if(p<0)

{

x=x+1;

p=p+2dy;

}

else

{

x=x+1;

y=y+1;

p=p+2\*(dy-dx);

}

Putpixel(x,y,7);

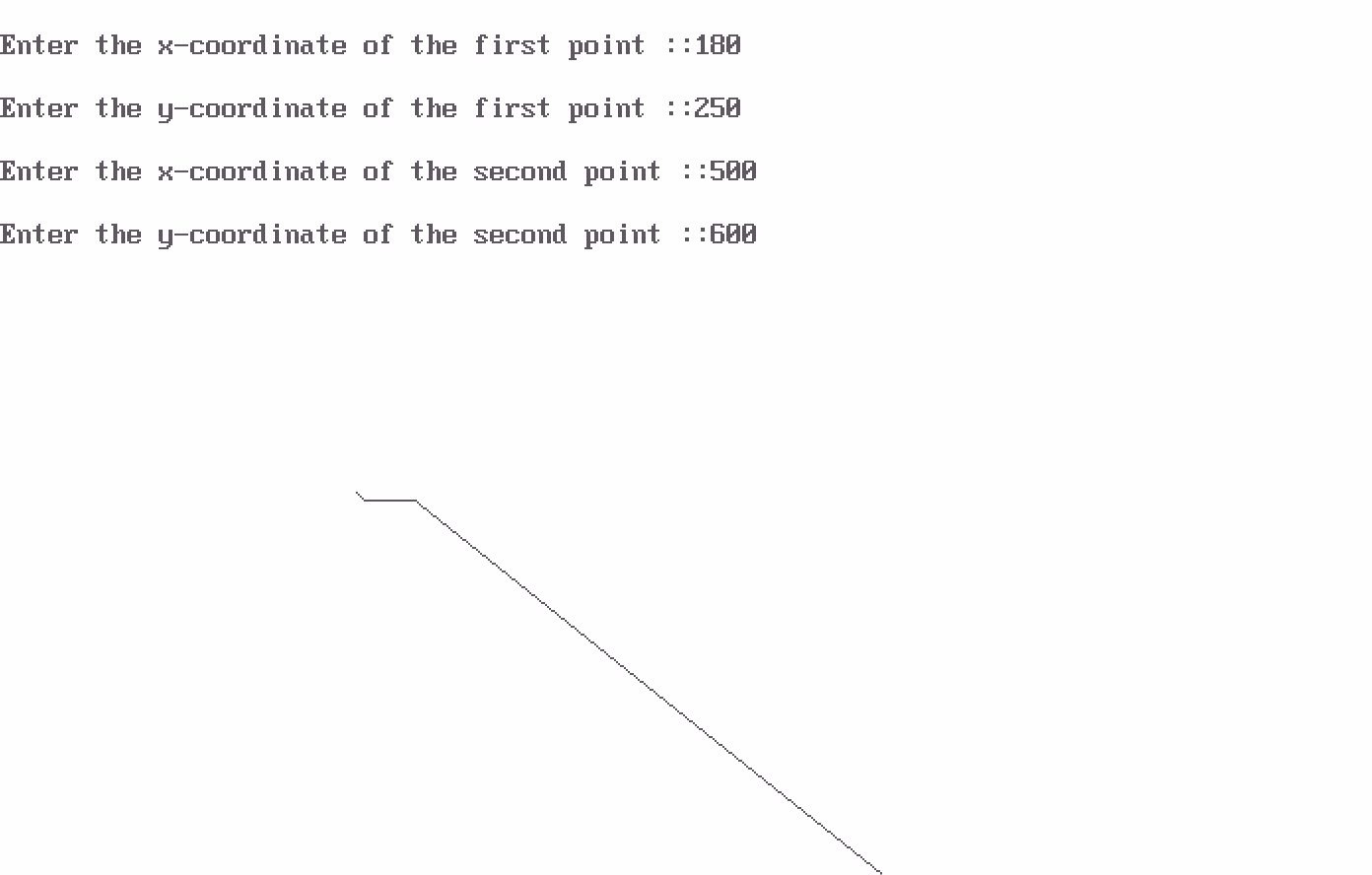
}

getch();

closegraph();

}

# OUTPUT



***ALGORITHM TO DRAW A LINE USING DDA ALGORITHM.***

1. Start.
2. Declare variables x,y,x1,y1,x2,y2,k,dx,dy,s,xi,yi and also declare gdriver=DETECT,gmode.
3. Initialise the graphic mode with the path location in TC folder.
4. Input the two line end-points and store the left end-points in (x1,y1).
5. Load (x1,y1) into the frame buffer;that is,plot the first point.put x=x1,y=y1.
6. Calculate dx=x2-x1 and dy=y2-y1.
7. If abs(dx) > abs(dy), do s=abs(dx).
8. Otherwise s= abs(dy).
9. Then xi=dx/s and yi=dy/s.
10. Start from k=0 and continuing till k<s,the points will be
    1. x=x+xi.
    2. y=y+yi.
11. Place pixels using putpixel at points (x,y) in specified colour.
12. Close Graph.
13. Stop.

**EXPERIMENT**

**Aim :- WAP TO DRAW A LINE USING DDA ALGORITHM.**

#include<stdio.h> #include<conio.h> #include<graphics.h> void main()

{

int x,y,x1,x2,y1,y2,k,dx,dy,s,xi,yi; int gdriver=DETECT,gmode;

initgraph(&gdriver,&gmode,"C:\\tc\\bgi:"); printf("enter first point"); scanf("%d%d",&x1,&y1);

printf("enter second point"); scanf("%d%d",&x2,&y2); x=x1;

y=y1; putpixel(x,y,7); dx=x2-x1; dy=y2-y1;

if(abs(dx)>abs(dy))

s=abs(dx);

else

s=abs(dy);

xi=dx/s; yi=dy/s; x=x1; y=y1;

putpixel(x,y,7); for(k=0;k<s;k++)

{

x=x+xi;

y=y+yi; putpixel(x,y,7);

}

getch(); closegraph();

}

# OUTPUT

***BRESENHAM’S ALGORITHM TO DRAW A CIRCLE*.**

1. Start.
2. Declare variables x,y,p and also declare gdriver=DETECT,gmode.
3. Initialise the graphic mode with the path location in TC folder.
4. Input the radius of the circle r.
5. Load x-0,y=r,initial decision parameter p=1-r.so the first point is (0,r).
6. Repeat Step 7 while (x<y) and increment x-value simultaneously. 7. If (p>0),do p=p+2\*(x-y)+1.
7. Otherwise p=p+2\*x+1 and y is decremented simultaneously.
8. Then calculate the value of the function circlepoints() with p.arameters (x,y).
9. Place pixels using putpixel at points (x+300,y+300) in specified colour in circlepoints() function shifting the origin to 300 on both x-axis and y-axis.
10. Close Graph.
11. Stop.

**Aim:- WAP TO DRAW A CIRCLE USING BRESENHAM’S ALGORITHM**.

#include<stdio.h> #include<conio.h> #include<graphics.h> void circlepoints(int,int); void main()

{

int x,y,p,r;

int gdriver=DETECT,gmode; initgraph(&gdriver,&gmode,"C:\\tc\\bgi:"); clrscr();

printf("enter the radius"); scanf("%d",&r); x=0;y=r;p=1-r; while(x<y)

{

x++;

if(p>0)

{

}

else

p=p+2\*(x-y)+1;

y--;

p=p+2\*x+1;

circlepoints(x,y);

}

getch(); closegraph();

}

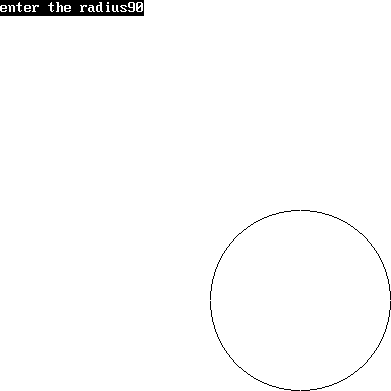
void circlepoints(int x,int y)

{

putpixel(x+300,y+300,8); putpixel(x+300,-y+300,8); putpixel(-x+300,y+300,8); putpixel(-x+300,-y+300,8); putpixel(y+300,x+300,8); putpixel(y+300,-x+300,8); putpixel(-y+300,x+300,8); putpixel(-y+300,-x+300,8);

}

# OUTPUT



***ALGORITHM TO DRAW AN ELLIPSE*.**

1. Start.
2. Initialize the graphic system using initgraph function.
3. Get the input of radius of major and minor arc from the user.
4. Store the values of major and minor arc in an another variable.
5. Square the values of major and minor arc.
6. Calculate decision parameter P = (square of minor axis – (square of major axis\*minor axis)

+ (0.25\* square of major axis).

1. Put the pixels symmetrically at = (0, length of minor axis).
2. while (2\*(square of minor axis\*x)<=2\*(square of major axis\*y)), repeat steps 9 to step 17.
3. increment x axis by 1.
4. If P < 0
5. new P = (P+( square of minor axis\* square of major axis)+ square of major axis)
6. Else
7. new P = (P+( square of minor axis\*x axis)-(2\*square of major axis\*y axis)+ square of minor axis).
8. Decrement y by 1.
9. End of step 10 if else structure.
10. Plot symmetric points of ellipse in each quadrant.
11. End of step 8 loop.
12. This will give us ellipse only across minor axis now to draw an ellipse across major axis we proceed further.
13. Get last point of ellipse in 1st quadrant.
14. Initialize e = square of (x axis+.5)
15. Initialize f = square of (y axis-1).
16. Decision parameter P1 = ((square of minor axis\*e)+( square of major axis\*f)-( square of minor axis\* square of major axis).
17. While y axis != 0 repeat steps 24 to step 32.
18. If P1>0
19. New P1 = (P1+ square of major axis-(2\* square of major axis\*x axis)).
20. Else
21. New P1 = (P1+(2\*square of minor axis\*(x axis+1))-(2\* square of major axis\*(y axis- 1))+square of major axis).
22. Increment x axis by 1.
23. End of step 25 if else structure
24. Decrement y axis by 1.
25. Plot symmetric point in all quadrants
26. End of step 23 while loop.
27. Close the graphic system.
28. Stop.

**Aim:- WAP TO DRAW AN ELLIPSE USING MID-POINT ELLIPSE DRAWING ALGORITHM.**

#include<stdio.h> #include<conio.h> #include<graphics.h> #include<math.h> void ellips(int x,int y);

void completellipse(int r,int g,int u,int v)

{

float s,k,e,f,x; double p1,p2; s=r;k=g; e=(pow((s+.5),2));

f=(pow((k-1),2));

p2=((u\*e)+(v\*f)-(u\*v)); ellips(s,k);

while(k>=0)

{

if(p2>0)

p2=(p2+v-(2\*v\*s));

else

{

}

k--;

p2=(p2+(2\*u\*(s+1))-(2\*v\*(k-1))+v);

s++;

ellips(s,k);

}

}

void main()

{

int gdriver=DETECT,gmode; int a,b,x,y;

long u,v,p1; initgraph(&gdriver,&gmode,"C:\\tc\\bgi::"); printf("\n enter the length of major axis:"); scanf("\t%d",&a);

printf("\n enter the length of minor axis:"); scanf("\t%d",&b);

x=0;

y=b; u=pow(b,2);

v=pow(a,2);

p1=(u-(v\*b)+(.25\*v)); ellips(x,y); while(2\*(u\*x)<=2\*(v\*y))

{

x++;

if(p1<0)

p1=(p1+(2\*u\*v)+v);

else

{

}

p1=(p1+(2\*u\*x)-(2\*v\*y)+u); y--;

ellips(x,y);

}

completellipse(x,y,u,v); getch();

closegraph();

}

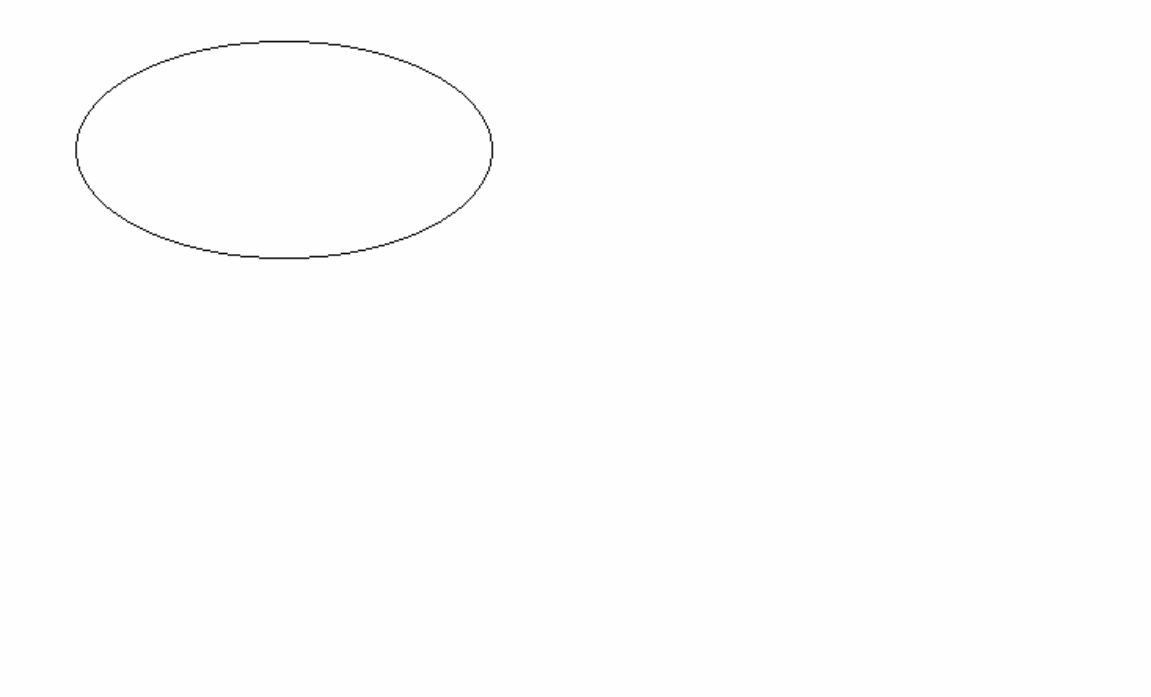
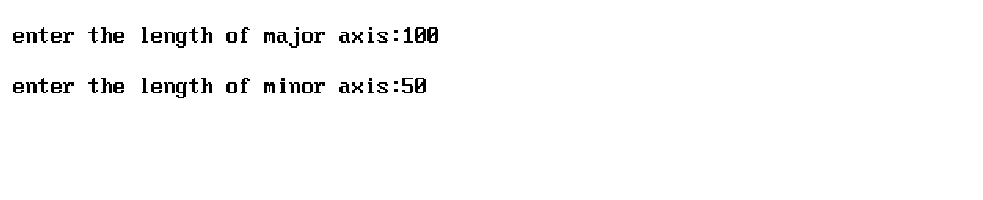
void ellips(int x,int y)

{

putpixel(x+200,y+200,8); putpixel(-x+200,y+200,8); putpixel(x+200,-y+200,8); putpixel(-x+200,-y+200,8);

}

# OUTPUT



***ALGORITHM TO CLIP A LINE.***

1. Start.
2. Initialize the graphic system using initgraph function.
3. Get the input of window co ordinates from the user and draw a window.
4. Get the input of line co ordinates from user and draw the line.
5. Calculate the region code of each end point of line using relation given in steps 6 to step
6. Let (x,y) be the co ordinates of end point of line and (xmin,ymin), (xmax,ymax) be co ordinates of world window
7. If y –ymax = +ve
8. MSB region code = 1.
9. Else MSB region code = 0.
10. If ymin – y = +ve
11. Region code = 1.
12. Else Region code = 0.
13. If x – xmax = +ve
14. Region code = 1.
15. Else Region code = 0.
16. If xmin – x = +ve
17. LSB Region code = 1.
18. Else LSB Region code = 0.
19. Calculate region code of both end points.
20. Logically and both region code.
21. If Logically anded result is = 0
22. Line is not a clipping candidate.
23. Else.
24. Line is a clipping candidate.
25. Calculate slope of line using formula slope=(y2-y1)/(x2-x1).
26. If line is to be horizontally clipped.
27. New y = ymin or ymax.
28. New x = x1 + ((new y - y1)/slope).
29. If line is to be vertically clipped.
30. New x = xmin or xmax.
31. New y = y1+slope\*(new x –x1).
32. Clip the lines from these intersection points.
33. Display the new line.
34. Close the graphic system.
35. Stop.

***WAP TO SHOW LINE CLIPPING*.**

#include<stdio.h> #include<conio.h> #include<graphics.h> #include<dos.h>

void storepoints(int,int,int,int,int,int,int[]); void main()

{

int gdriver=DETECT,gmode;

int x1,x2,y1,y2,xmax,ymax,xmin,ymin,a[10],b[10],xi1,xi2,yi1,yi2,flag=0; float m;

int i; clrscr();

printf("output"); printf("\n");

printf("enter the value of x1,y1,x2,y2: >"); scanf("%d%d%d%d",&x1,&y1,&x2,&y2); printf("enter the value of xmax,ymax,xmin,ymin:"); scanf("%d%d%d%d",&xmax,&ymax,&xmin,&ymin); storepoints(x2,y2,ymin,ymax,xmax,xmin,b); for(i=1;i<=4;i++)

{

if(a[i]\*b[i]==0)

flag=1;

else

}

flag=0;

if(flag==1)

{

m=(y2-y1)/(x2-x1); xi1=x1;

yi1=y1;

}

if(a[1]==1)

{

}

else

{

yi1=ymax; xi1=x1+((1/m)\*(yi1-y1));

if(a[2]==1)

{

yi1=ymin; xi1=x1+((1/m)\*(yi1-y1));

}

}

if(a[3]==1)

{

xi1=xmax; yi1=y1+(m\*(xi1-x1));

}

if(a[4]==1)

{

}

else

xi1=xmin; yi1=y1+(m\*(xi1-x1));

if(b[1]==1)

{

}

else

yi2=ymax; xi2=x2+((1/m)\*(yi2-y2));

if(b[2]==1)

{

}

else

yi2=ymin; xi2=x2+((1/m)\*(yi2-y2));

if(b[3]==1)

{

clrscr();

}

else

xi2=xmax; yi2=y2+((1/m)\*(xi2-x2));

if(b[4]==1)

{

xi2=xmin; yi2=y2+(m\*(xi2-x2));

}

initgraph(&gdriver,&gmode,"c://tc//bgi:"); rectangle(xmin,ymin,xmax,ymax); line(x1,y1,x2,y2);

delay(5000); closegraph(); clrscr();

initgraph(&gdriver,&gmode,"c://tc//bgi:"); line(xi1,yi1,xi2,yi2); rectangle(xmin,ymin,xmax,ymax);

if(flag==0)

{

printf("\n no clipping is required");

}

getch(); closegraph();

}

void storepoints(int x1,int y1,int ymax,int xmax,int xmin,int ymin,int c[10])

{

if((y1-ymax)>0)

c[1]=1;

else

c[1]=0;

if((ymin-y1)>0)

c[2]=1;

else

c[2]=0;

if((x1-xmax)>0)

c[3]=1;

else

c[3]=0;

if((xmin-x1)>0)

c[4]=1;

else

}

c[4]=0;

# OUTPUT

enter the value of x1,y1,x2,y2: >10 10

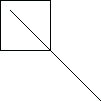
100

100

enter the value of xmax,ymax,xmin,ymin50 50

0

0





# WAP TO ROTATE A TRIANGLE ABOUT ORIGIN.

#include<iostream.h> #include<conio.h> #include<graphics.h> #include<process.h> #include<math.h> void main()

{

clrscr();

int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi");

int x,y,x1,a[3][3];

double b[3][3],c[3][3];

cout<<"\n Enter Ist coordinates of triangle:"; cin>>a[0][0]>>a[1][0];

cout<<"\n Enter 2nd coordinates of triangle:"; cin>>a[0][1]>>a[1][1];

cout<<"\n Enter 3rd coordinates of triangle:"; cin>>a[0][2]>>a[1][2];

line(a[0][0],a[1][0],a[0][1],a[1][1]);

line(a[0][1],a[1][1],a[0][2],a[1][2]);

line(a[0][0],a[1][0],a[0][2],a[1][2]);

getch(); cleardevice();

cout<<"\n Enter angle of rotation:\n"; cin>>x;

b[0][0]=b[1][1]=cos((x\*3.14)/180);

b[0][1]=-sin((x\*3.14)/180);

b[1][0]=sin((x\*3.14)/180); b[2][2]=1; b[2][0]=b[2][1]=b[0][2]=b[1][2]= 0;

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{ c[i][j]=0;

for (int k=0; k<3;k++)

{

c[i][j]+=a[i][k]\*b[k][j];

} x1=(c[i][j]+0.5);

a[i][j]=x1;

}

}

cout<<"\n Triangle after rotation is:\n" ;

line(a[0][0],a[1][0],a[0][1],a[1][1]);

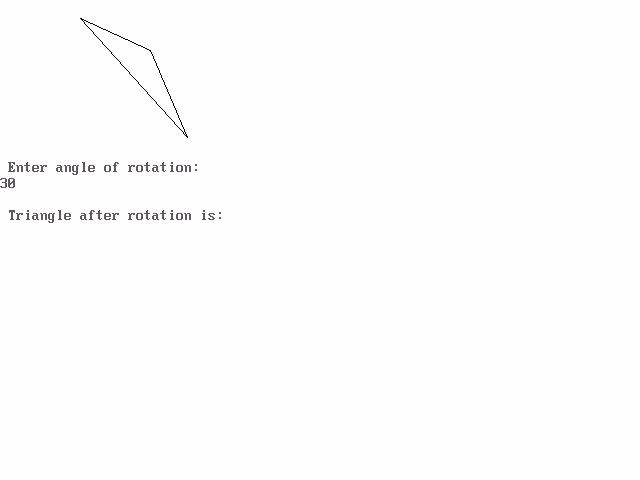
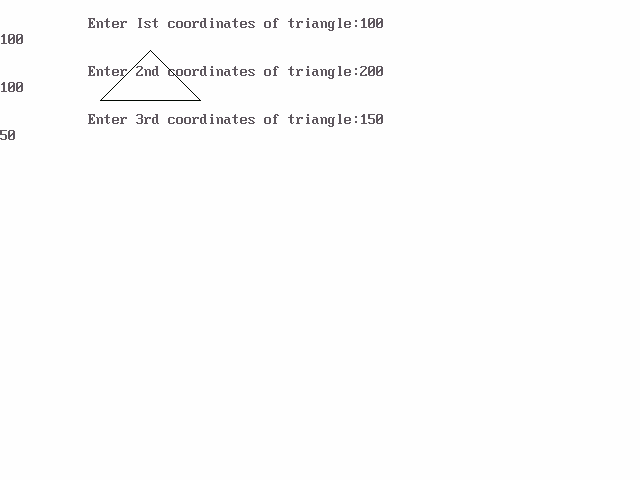
line(a[0][1],a[1][1],a[0][2],a[1][2]);

line(a[0][0],a[1][0],a[0][2],a[1][2]);

getch(); closegraph();

}

***OUTPUT***



***PROGRAM T O SCALE THE TRIANGLE***

#include<iostream.h> #include<conio.h> #include<graphics.h>O void main()

{

int gd=DETECT,gm; initgraph(&gd, &gm,""); cleardevice();

int x1,y1,x2,y2,x3,y3,x4,y4; float sx,sy;

cout<<"Enter the first coordinates of triangle\n"; cin>>x1>>y1;

cout<<"Enter the second coordinates of triangle\n"; cin>>x2>>y2;

cout<<"Enter the third coordinates of triangle\n"; cin>>x3>>y3;

int poly[8]={x1,y1,x2,y2,x3,y3,x1,y1}; cleardevice();

drawpoly(4,poly); getch();

cout<<"Enter the scaling factors\n"; cin>>sx>>sy;

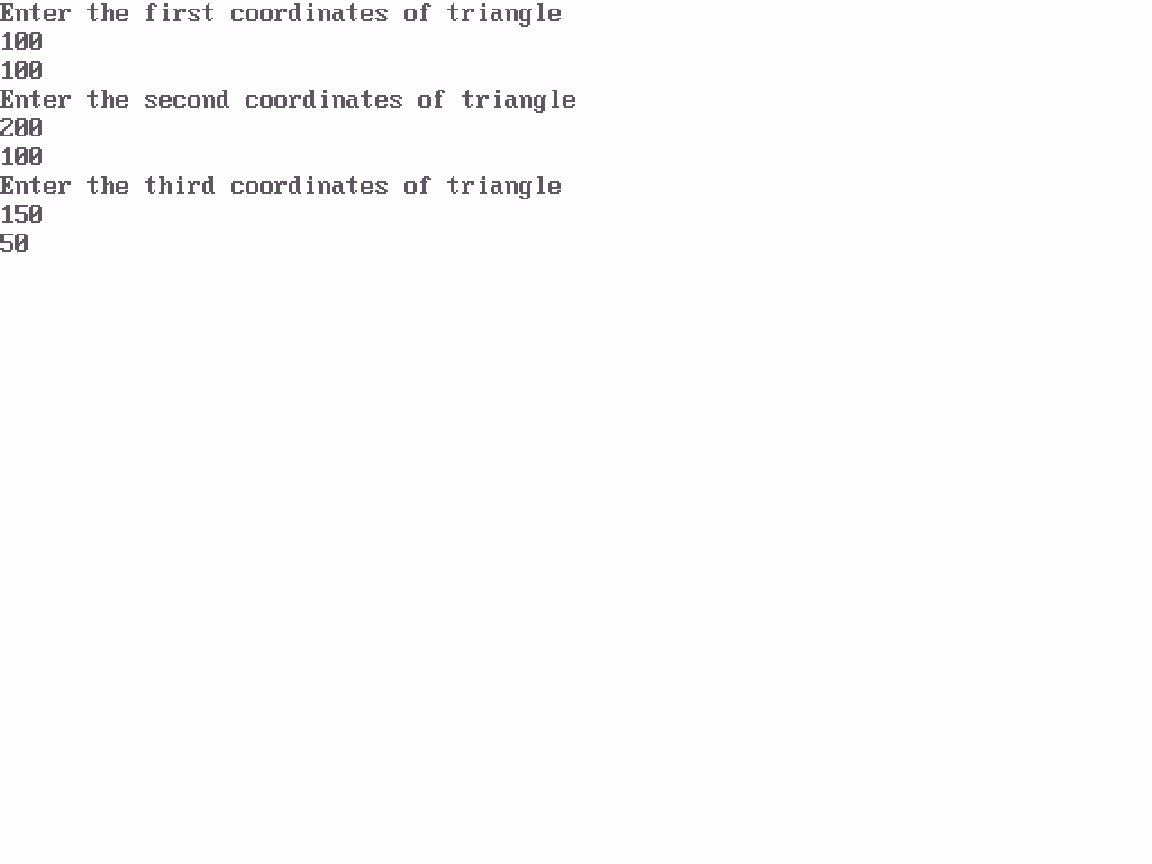
x4=sx\*x1-x1; y4=sy\*y1-y1;

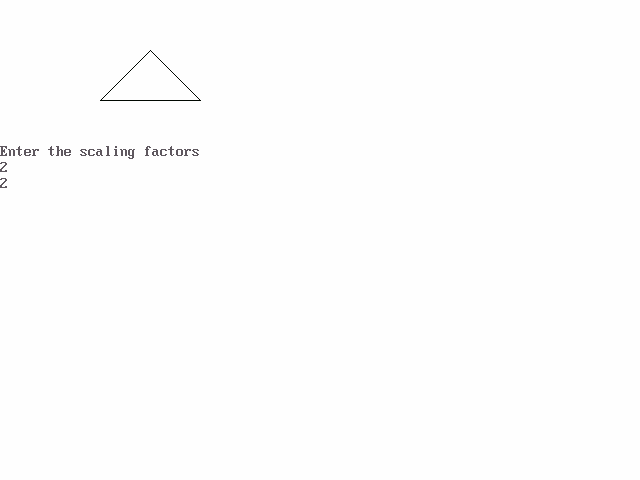
x1=sx\*x1-x4; y1=sy\*y1-y4; x2=sx\*x2-x4;

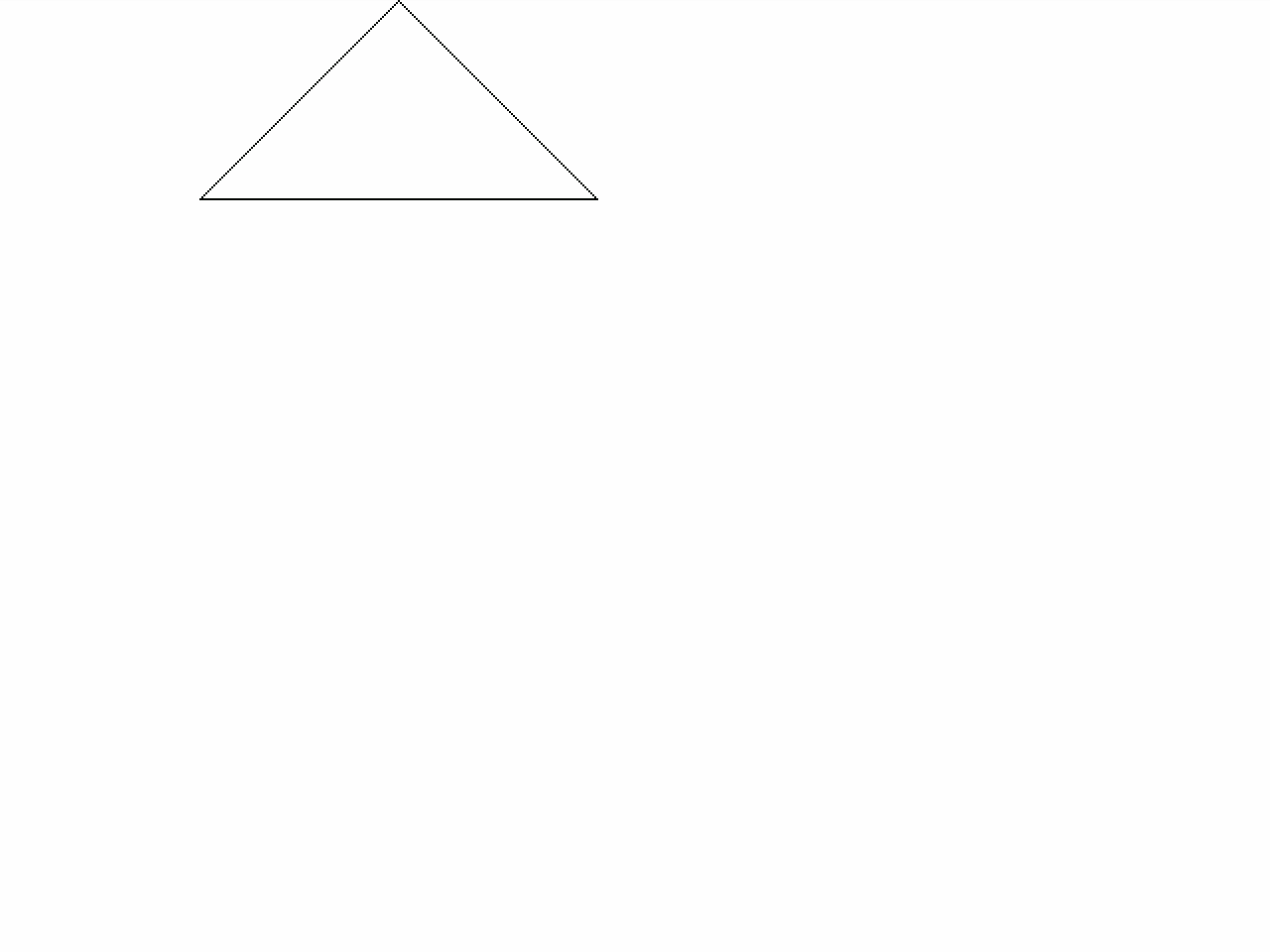
y2=sy\*y2-y4; x3=sx\*x3-x4; y3=sy\*y3-y4; poly[0]=x1; poly[1]=y1; poly[2]=x2; poly[3]=y2; poly[4]=x3; poly[5]=y3; poly[6]=x1; poly[7]=y1; getch(); cleardevice(); drawpoly(4,poly); getch(); closegraph();

}

***OUTPUT***







***PROGRAM TO TRANSLATE A TRIANGLE***

#include<iostream.h> #include<conio.h> #include<graphics.h> #include<process.h> #include<math.h>

void main()

{

clrscr();

int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi");

int x,y,x1,y1,x2,y2,x3,y3;

cout<<"\n cin>>x1>>y1;

cout<<"\n cin>>x2>>y2;

cout<<"\n cin>>x3>>y3;

Enter Ist coordinates of triangle:";

Enter 2nd coordinates of triangle:";

Enter 3rd coordinates of triangle:";

cleardevice(); line(x1,y1,x2,y2);

line(x2,y2,x3,y3);

line(x1,y1,x3,y3); getch(); cleardevice();

cout<<"\n Enter translatio factors :\n"; cin>>x>>y;

x1-=x;

y1-=y;

x2-=x;

y2-=y;

x3-=x;

y3-=y;

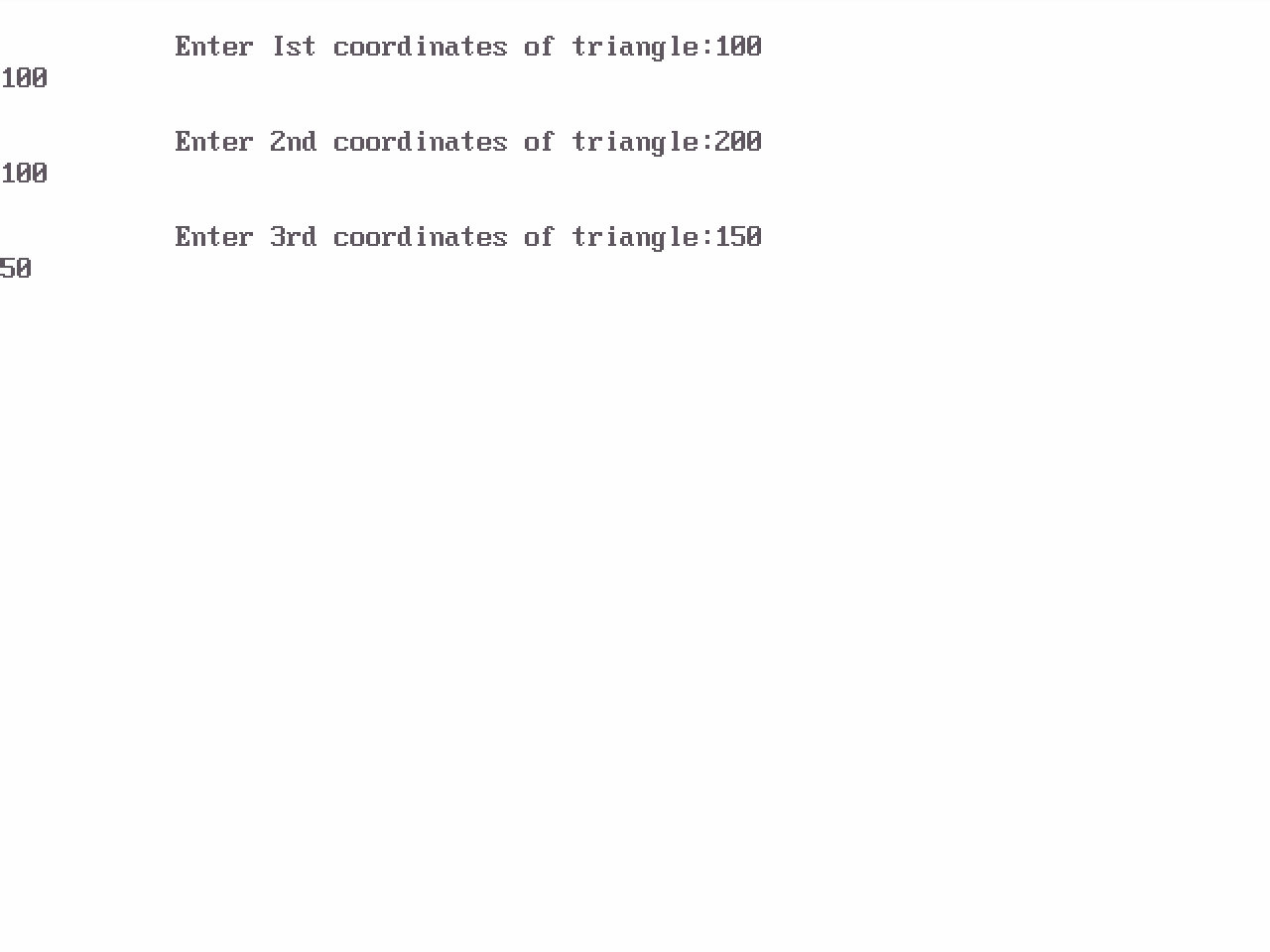
cleardevice(); line(x1,y1,x2,y2);

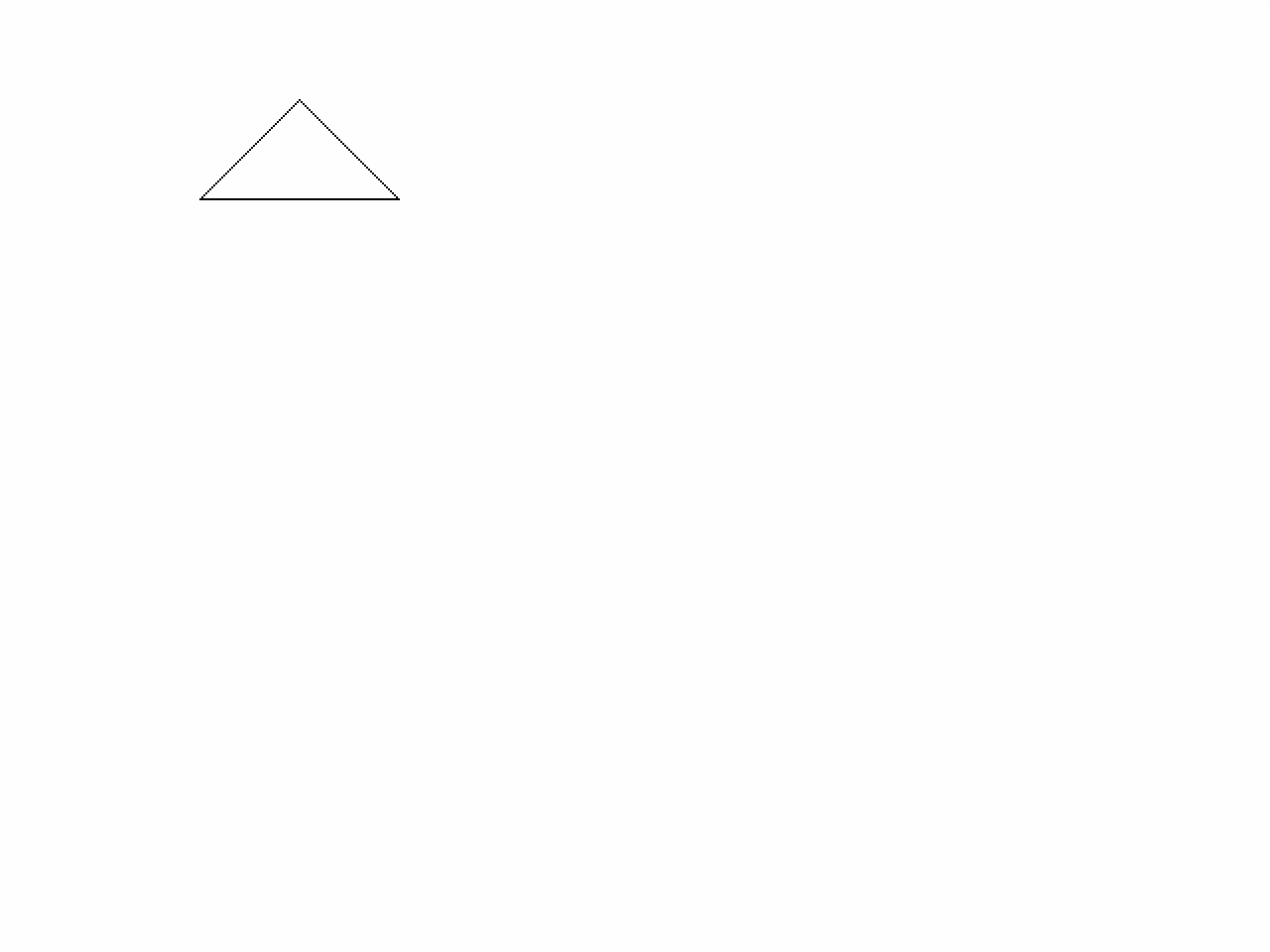
line(x2,y2,x3,y3);

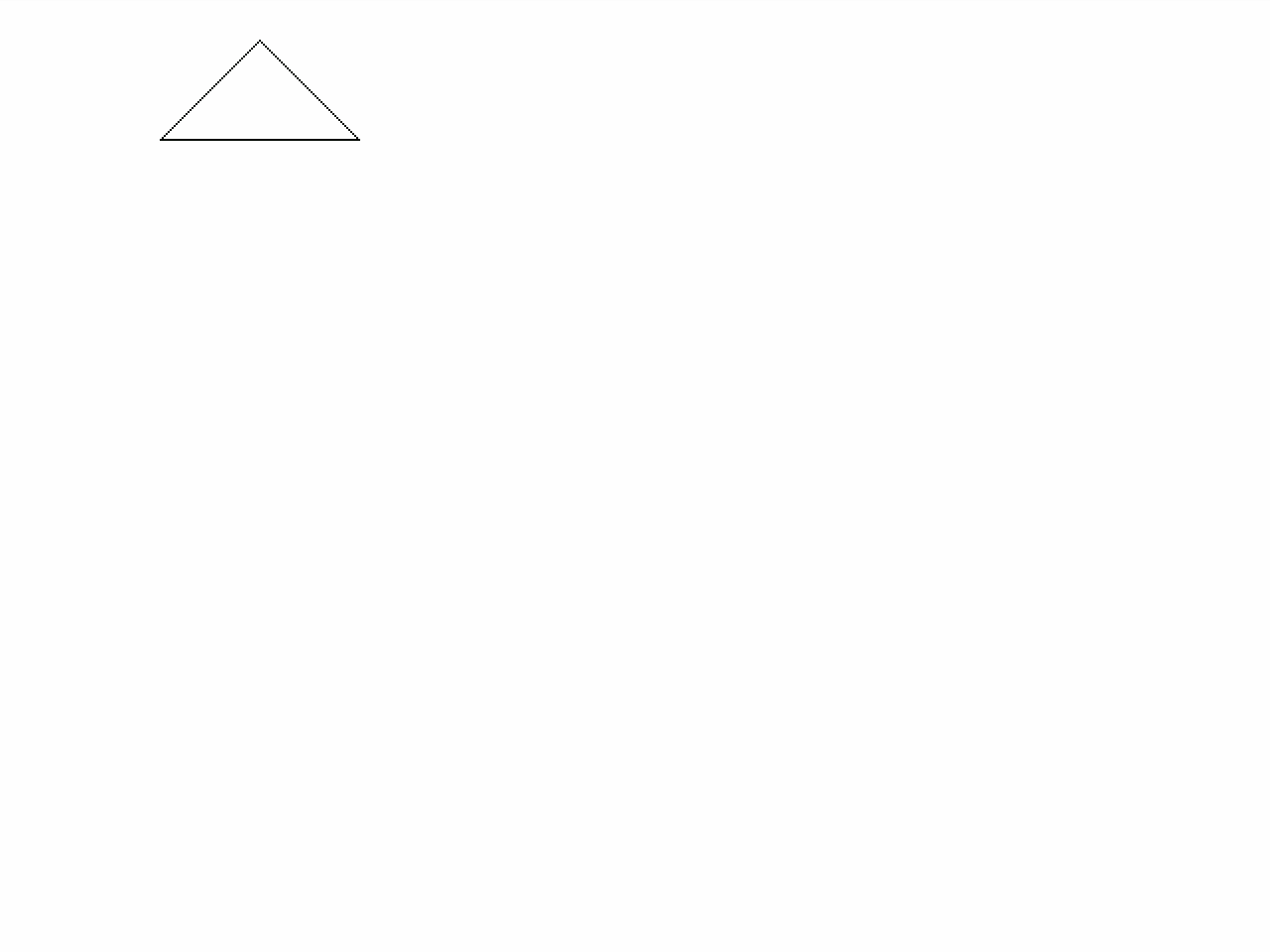
line(x1,y1,x3,y3); getch(); closegraph();

}

***OUTPUT***







***PROGRAM TO ROTATE A POINT ABOUT A POINT***

#include<iostream.h> #include<conio.h> #include<graphics.h> #include<math.h> #include<dos.h>

void main()

{

clrscr();

int gm,gd=DETECT; initgraph(&gd,&gm,"");

int h,k,x1,y1,x2,y2,x3,y3; float t;

cout<<" OUTPUT"<<endl;

cout<<"Enter the coordinates of point"<<endl; cin>>x2>>y2;

putpixel(x2,y2,2);

cout<<"Enter the coordinates of point around which rotation is done"<<endl; cin>>h>>k;

putpixel(h,k,2);

cout<<"Enter the angle for rotation"<<endl; cin>>t;

cleardevice(); x1=(h\*cos(t))-(k\*sin(t));

y1=(h\*sin(t))+(k\*cos(t)); x3=x1+x2-h;

y3=y1+y2-k;

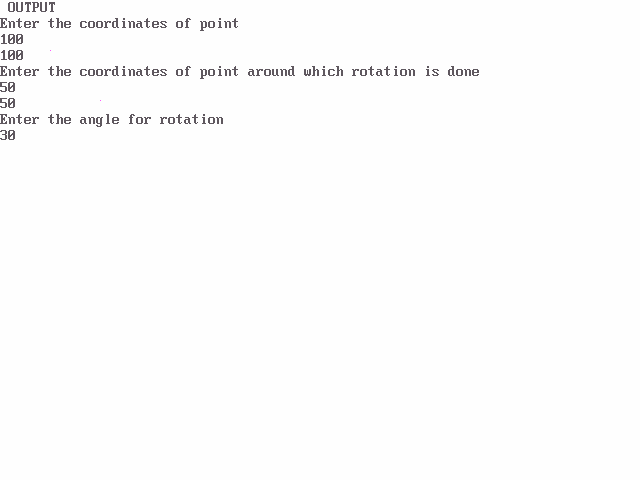
cout<<"Point after rotation is:";

putpixel(x3,y3,2);

getch(); closegraph();

}

***OUTPUT***





***PROGRAM TO ROTATE A POINT ABOUT ORIGIN***

#include<iostream.h> #include<conio.h> #include<graphics.h> #include<math.h> #include<dos.h>

void main()

{

clrscr();

int gm,gd=DETECT; initgraph(&gd,&gm,"");

int h,k,x1,y1,x2,y2,x3,y3; float t;

cout<<" OUTPUT"<<endl;

cout<<"Enter the coordinates of point"<<endl; cin>>x2>>y2;

putpixel(x2,y2,2);

cout<<"Enter the angle for rotation"<<endl; cin>>t;

cleardevice();

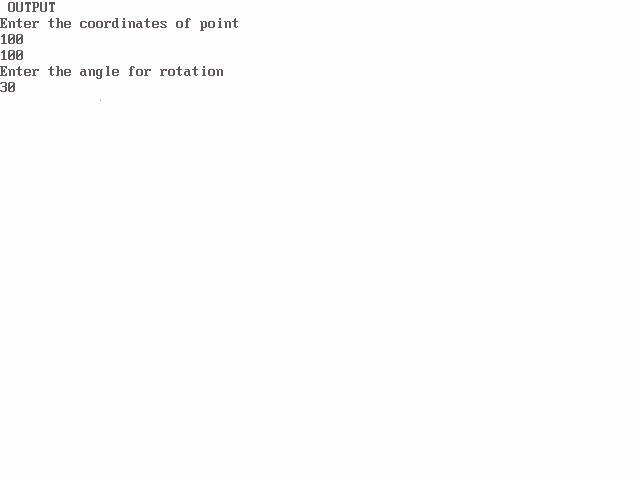
x1=int(x2\*cos(t\*3.14/180))-(y2\*sin(t\*3.14/180)); y1=int(x2\*sin(t\*3.14/180))+(y2\*cos(t\*3.14/180)); cout<<"Point after rotation is:";

putpixel(x1,y1,2);

getch(); closegraph();

}

***OUTPUT***





***PROGRAM TO REFLECT A TRIANGLE***

#include<iostream.h> #include<conio.h> #include<graphics.h> #include<process.h> #include<math.h> void main()

{

clrscr();

int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi");

int x,y,x1,a[3][3];

double b[3][3],c[3][3];

cout<<"\n Enter Ist coordinates of triangle:"; cin>>a[0][0]>>a[1][0];

cout<<"\n Enter 2nd coordinates of triangle:"; cin>>a[0][1]>>a[1][1];

cout<<"\n Enter 3rd coordinates of triangle:"; cin>>a[0][2]>>a[1][2];

cout<<"\n Enter 1. for reflection in x-axis:\n"; cout<<"\n Enter 2. for reflection in y-axis:\n"; cout<<"\n Enter 3. for reflection in both the axis:\n"; cin>>x;

cleardevice(); line(320,0,320,479); line(0,240,639,240);

line(a[0][0],a[1][0],a[0][1],a[1][1]);

line(a[0][1],a[1][1],a[0][2],a[1][2]);

line(a[0][0],a[1][0],a[0][2],a[1][2]);

switch(x)

{

case 1:b[0][0]=640-a[0][0];

b[0][1]=640-a[0][1];

b[0][2]=640-a[0][2];

b[1][0]=a[1][0];

b[1][1]=a[1][1];

b[1][2]=a[1][2]; line(320,0,320,479); line(0,240,639,240);

line(b[0][0],b[1][0],b[0][1],b[1][1]);

line(b[0][1],b[1][1],b[0][2],b[1][2]);

line(b[0][0],b[1][0],b[0][2],b[1][2]);

getch(); break;

case 2:b[1][0]=480-a[1][0];

b[1][1]=480-a[1][1];

b[1][2]=480-a[1][2];

b[0][0]=a[0][0];

b[0][1]=a[0][1];

b[0][2]=a[0][2]; line(320,0,320,479); line(0,240,639,240);

line(b[0][0],b[1][0],b[0][1],b[1][1]);

line(b[0][1],b[1][1],b[0][2],b[1][2]);

line(b[0][0],b[1][0],b[0][2],b[1][2]);

getch(); break;

case 3: b[0][0]=640-a[0][0];

b[0][1]=640-a[0][1];

b[0][2]=640-a[0][2];

b[1][0]=a[1][0];

b[1][1]=a[1][1];

b[1][2]=a[1][2]; line(320,0,320,479); line(0,240,639,240);

line(b[0][0],b[1][0],b[0][1],b[1][1]);

line(b[0][1],b[1][1],b[0][2],b[1][2]);

line(b[0][0],b[1][0],b[0][2],b[1][2]); b[1][0]=480-a[1][0];

b[1][1]=480-a[1][1];

b[1][2]=480-a[1][2];

b[0][0]=a[0][0];

b[0][1]=a[0][1];

b[0][2]=a[0][2]; line(320,0,320,479); line(0,240,639,240);

line(b[0][0],b[1][0],b[0][1],b[1][1]);

line(b[0][1],b[1][1],b[0][2],b[1][2]);

line(b[0][0],b[1][0],b[0][2],b[1][2]);

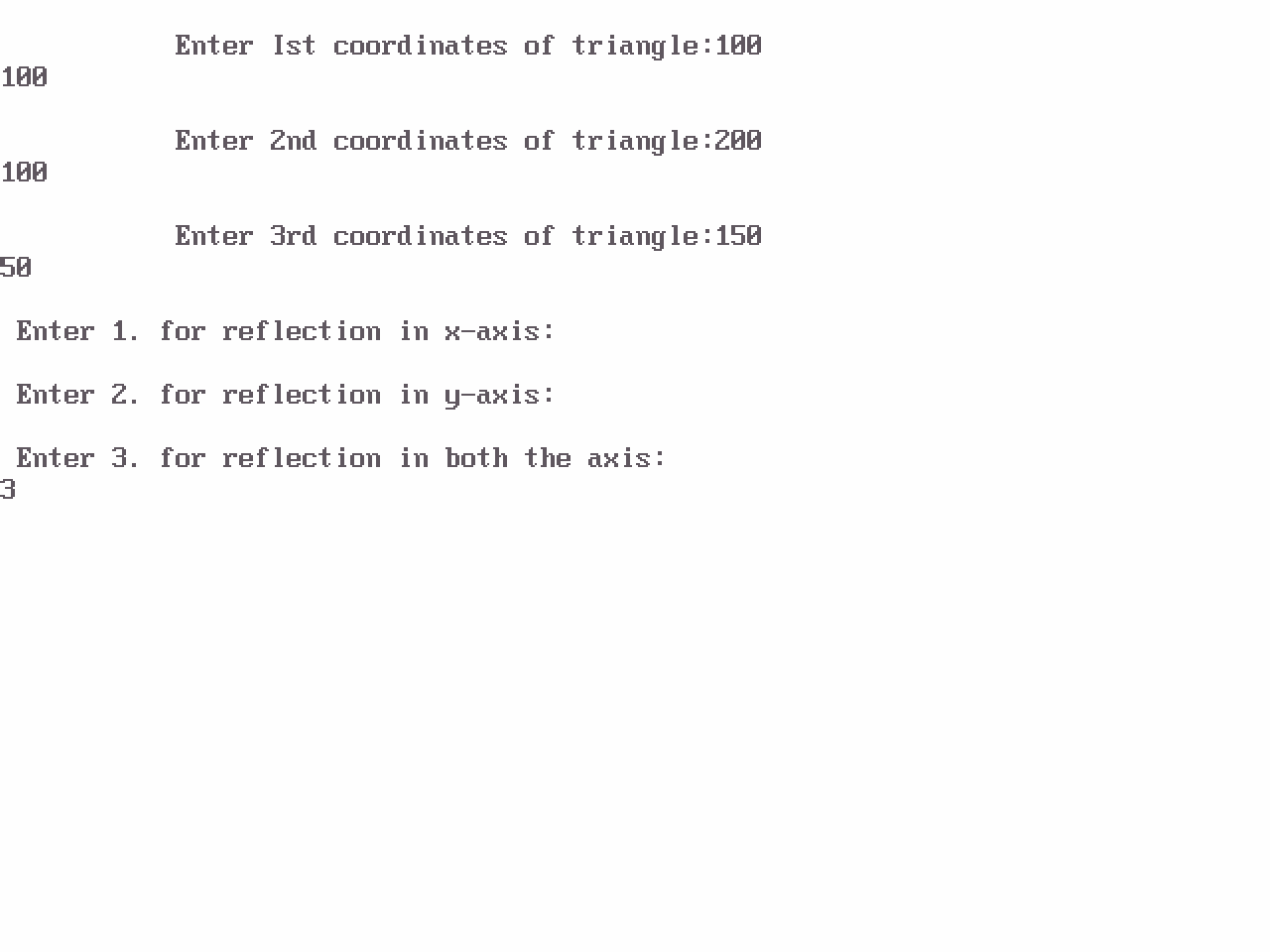
getch(); break;

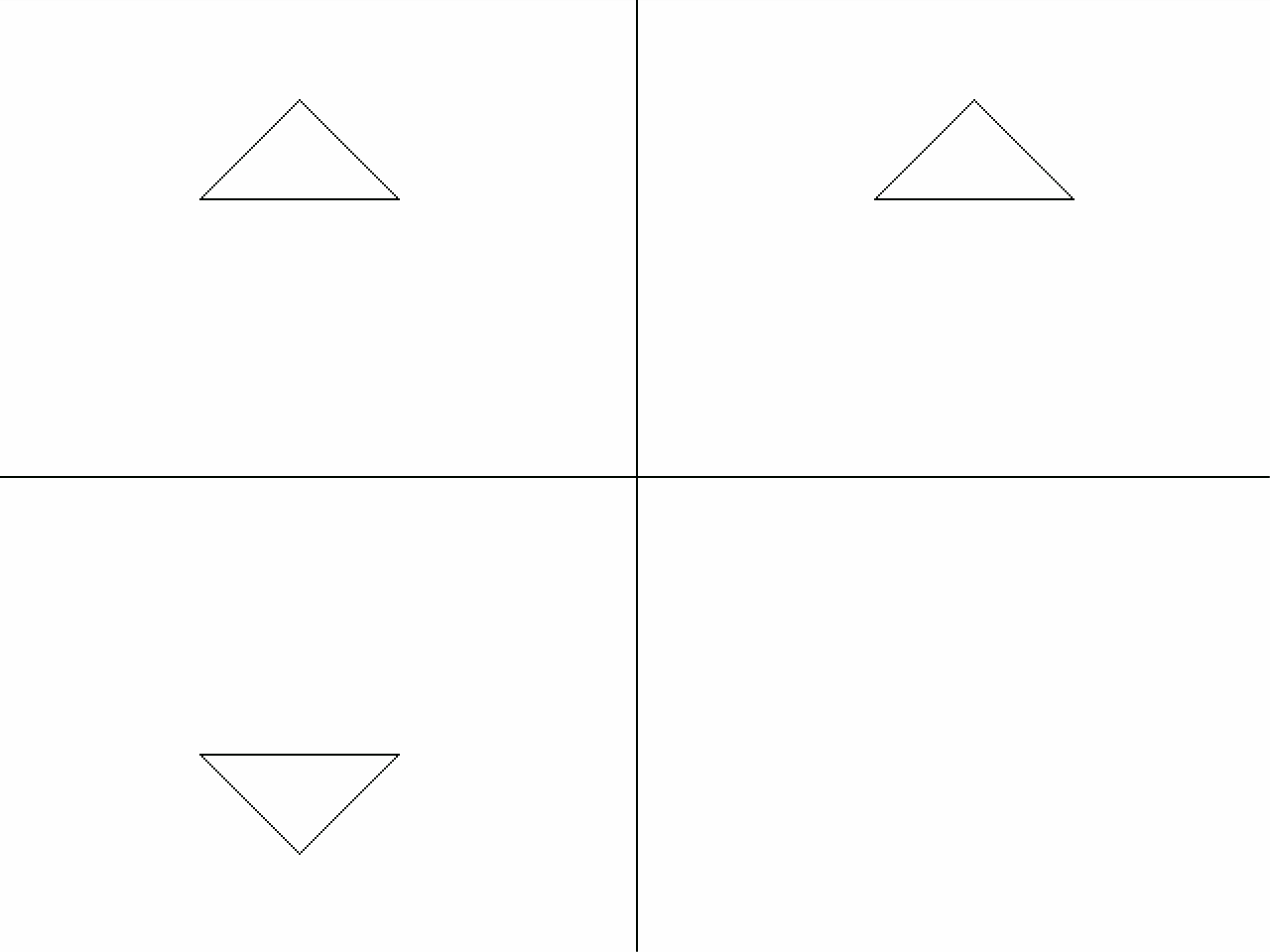
}

getch(); closegraph();

}

***OUTPUT***





***PROGRAM TO DRAW A HUT USING SIMPLE GRAPHIC FUNCTIONS***

#include<conio.h> #include<iostream.h> #include<graphics.h> #include<math.h> #include<dos.h> #include<process.h> void main()

{

int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi"); line(100,100,150,50);

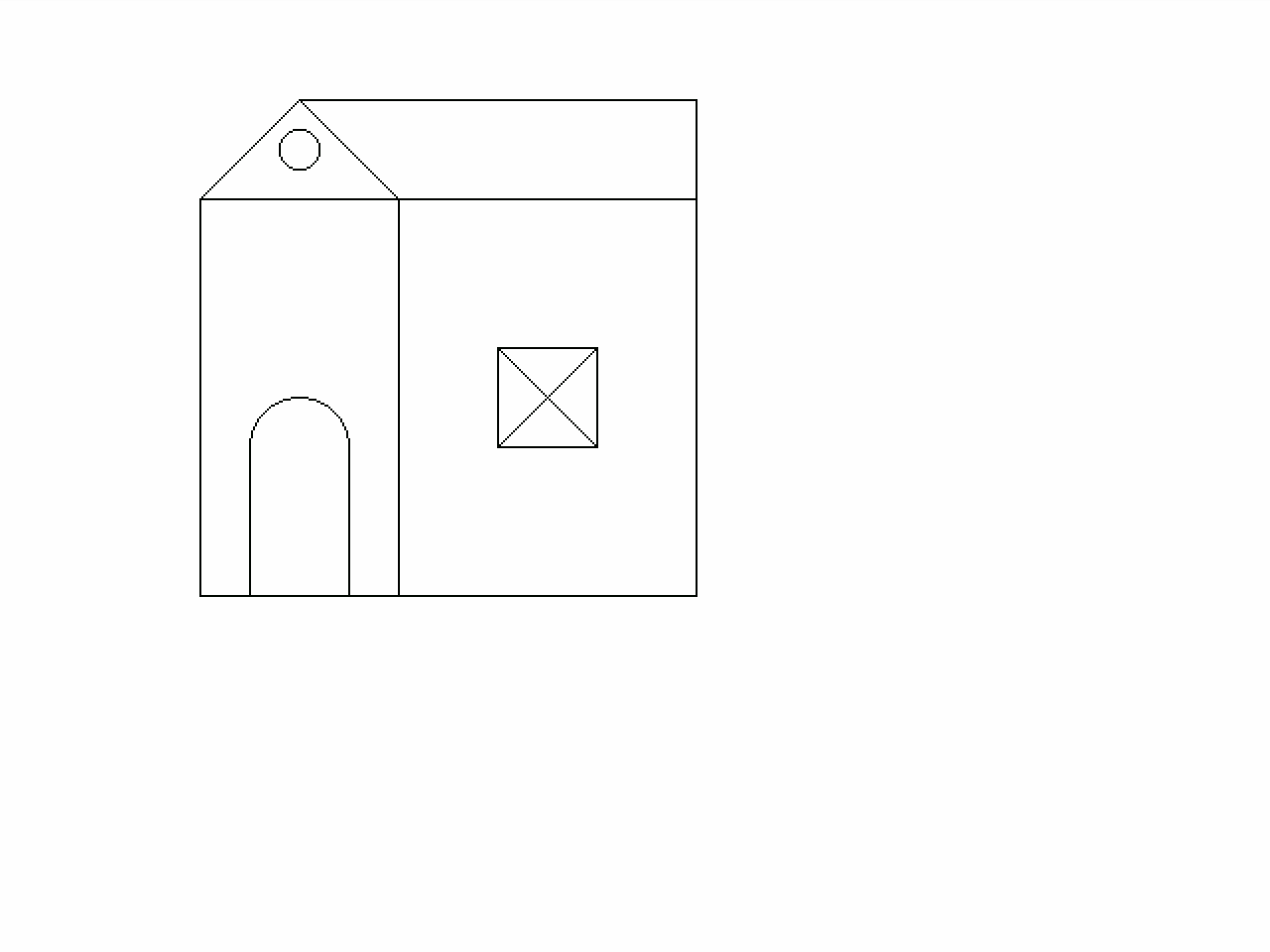
line(150,50,200,100); line(100,100,200,100); line(150,50,350,50); line(200,100,350,100); line(350,50,350,100);

circle(150,75,10); rectangle(100,100,200,300); rectangle(200,100,350,300); rectangle(250,175,300,225); line(250,175,300,225); line(300,175,250,225); line(125,300,125,225); line(175,300,175,225); arc(150,225,0,180,25);

getch(); closegraph();

}

***OUTPUT***



***PROGRAM TO FILL A POLYGON***

#include<conio.h> #include<iostream.h> #include<graphics.h> #include<math.h> #include<dos.h> #include<process.h> void main()

{

int graphdriver=DETECT,graphmode; initgraph(&graphdriver,&graphmode,"...\\bgi"); int p=1,x;

int a[12]={100,100,150,150,200,100,200,200,100,200,100,100};

drawpoly(6,a);

for(int i=100;i<200;i++)

{ p=1;

for(int j=100;j<=200;j++)

{

x=getpixel(j,i);

for(int d=0;d<11;d++)

{

if(j==a[d]&&i==a[d+1] ) break;

else

{

if(x>0&&d==10) p++;

if(p%2==0)

putpixel(j,i,4);

}}}}

getch();

closegraph();

}

***OUTPUT***

